Project report

Bitcoin prediction model

Chantoiseau Sacha
Cherigui Allah Eddine
Louis Malmassary

January 6, 2025

Figure 1: University Logo

Contents

1 Introduction

The goal of this project is to develop a predictive model for Bitcoin prices using machine learning techniques. Bitcoin, being a highly volatile cryptocurrency, presents a challenging task for accurate price prediction. The primary objective is to build a model that can predict future Bitcoin prices based on historical data, thereby providing insights and aiding in decision-making for investors and traders.

2 Data Set Description

The data set used for this project consists of historical Bitcoin prices obtained from a publicly available source. The data includes daily prices of Bitcoin in USD, along with other relevant features such as the date. The data preprocessing steps involved the following:

- Reading the raw data from a CSV file.
- Converting the date column to a datetime format.
- Extracting additional features such as year and month from the date.
- Normalizing the relevant columns to ensure that the data is on a similar scale.
- Aggregating the data on a monthly basis to reduce noise and capture long-term trends.

The preprocessed data is then saved to a new CSV file for further analysis and model training.

3 Methodology

The methodology followed in this project involves several key steps:

3.1 Model Selection

Given the time series nature of the data, an LSTM (Long Short-Term Memory) neural network was chosen for the prediction task. LSTM models are well-suited for time series data as they can capture temporal dependencies and patterns.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

def build_lstm_model(input_shape):
    model = Sequential([
        LSTM(50, input_shape=input_shape, return_sequences=True),
        Dropout(0.2),
        LSTM(50, return_sequences=False),
        Dropout(0.2),
        Dense(1)  # Sortie unique
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

Figure 2: LSTM Model Architecture

3.2 Data Preparation

The preprocessed data is divided into time windows to create input sequences for the LSTM model. Each time window consists of a fixed number of past observations used to predict the next value in the sequence. The data is then split into training and testing sets to evaluate the model's performance.

We use the loadAndPreprocessData function to load a CSV file, process the date column to extract the year and month, normalize specific financial columns with MinMaxScaler, and group the data by year and month, calculating the mean for each group. It returns the monthly aggregated data with normalized values, indexed by date.

```python
def load_and_preprocess_data(file_path):
    df = pd.read_csv(file_path)
    df['Date'] = pd.to_datetime(df['Date'])
    df['Year'] = df['Date'].dt.year
    df['Month'] = df['Date'].dt.month

    # Normalisation
    scaler = MinMaxScaler(feature_range=(-1, 1))
    cols_to_normalize = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
    df[cols_to_normalize] = scaler.fit_transform(df[cols_to_normalize])

    # Regroupement mensuel
    monthly_data = df.groupby(['Year', 'Month'])[cols_to_normalize].mean().reset_index()
    monthly_data['Date'] = pd.to_datetime(
        monthly_data['Year'].astype(str) + '-' + monthly_data['Month'].astype(str) + '-01'
    )
    monthly_data.set_index('Date', inplace=True)
    return monthly_data
```

Figure 3: Data Preparation Process

3.3 Model Training

The LSTM model is built using the Keras library with TensorFlow as the back-

end. The model consists of two LSTM layers and two Dropout layers to prevent overfitting. The model is compiled with the Adam optimizer and mean squared error loss function. The training process involves fitting the model to the training data and saving the trained model and test data for evaluation.

```python
def evaluate_model(model_path, X_test_path, y_test_path):
    model = load_model(model_path)
    X_test = np.load(X_test_path)
    y_test = np.load(y_test_path)

    # Faire des prédictions
    y_pred = model.predict(X_test)

    # Visualiser les résultats
    plt.figure(figsize=(12, 6))
    plt.plot(y_test, label='Réel', color='blue')
    plt.plot(y_pred, label='Prédit', color='orange')
    plt.title("Comparaison des valeurs réelles et prédites")
    plt.xlabel("Index")
    plt.ylabel("Prix normalisé")
    plt.legend()
    plt.grid()
    plt.savefig("img/predictions_plot.png")  # Sauvegarde
    plt.close()  # Fermer la figure pour éviter les problèmes
```

Figure 4: Model Evaluation

```python
def train_model(data_path, model_save_path, X_test_save_path, y_test_save_path):
    data = pd.read_csv(data_path, index_col="Date")

    # Création des fenêtres temporelles
    window_size = 3
    cols_to_use = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
    X, y = create_time_windows(data, window_size, cols_to_use)

    # Division des ensembles
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Construire et entraîner le modèle
    model = build_lstm_model(input_shape=(window_size, len(cols_to_use)))
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, batch_size=32)

    # Sauvegarder le modèle et les ensembles
    model.save(model_save_path)
    np.save(X_test_save_path, X_test)
    np.save(y_test_save_path, y_test)
```

Figure 5: Training Process

4 Results

4.1 Model Performance

The performance of the model was evaluated using the test data. The primary metric used for evaluation was the mean squared error (MSE) between the predicted and actual Bitcoin prices. The results indicate that the model was able to capture the general trend of Bitcoin prices, although there were some deviations.

4.2 Graphical Visualization

The real vs. predicted Bitcoin prices are visualized in the plot below. The plot provides a clear comparison between the actual prices and the model's predictions.

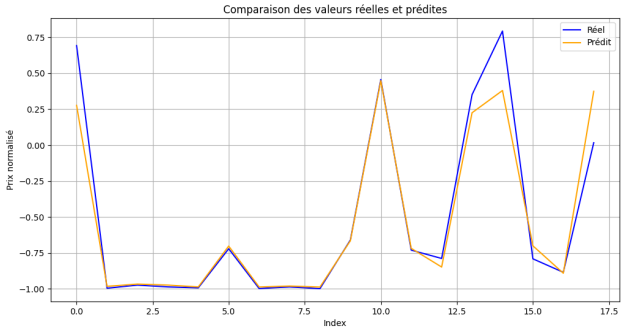

Figure 6: Real vs. Predicted Bitcoin Prices

5 Conclusions and Perspectives

5.1 Summary of Findings

The LSTM model developed in this project was able to predict Bitcoin prices with a reasonable degree of accuracy. The model captured the overall trend of Bitcoin prices, although there were some discrepancies between the predicted and actual prices.

5.2 Future Work

Future work could involve exploring different model architectures and hyperparameters to improve the prediction accuracy. Additionally, incorporating more features such as trading volume and market sentiment could potentially enhance the model's performance.

6 Team Presentation

6.1 Chantoiseau Sacha

Role: Data Preprocessing and Model Training

6.2 Cherigui Allah Eddine

Role: Model Evaluation and Results Visualization

6.3 Louis Malmassary

Role: Report Writing and Project Coordination